**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

“___”_____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

зі спеціальності **123 «Комп'ютерна інженерія»**

на тему: Events-додаток мовою програмування Java

Виконав: студент IV курсу, групи КВ-72

Панков Тимур Спартакович _____ _____
(підпис)

Керівник доц. каф. Потапова К. Р._____ _____
(підпис)

Консультант з нормоконтролю, доц. каф. СПСКС, к.т.н. Клятченко Я.М._____
(підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___»_____2021 р.

ЗАВДАННЯ
на дипломний проєкт студента
Панкова Тимура Спартаковича

1. Тема проєкту «Events-додаток мовою програмування Java», керівник проєкту Потапова К.Р., старший викладач, затверджені наказом по університету N1331-С від «25» травня 2021 р. №

2. Термін подання студентом проєкту: 25 травня 2021 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих додатків з додавання та пошуку заходів;
   - аналіз операційної системи Android;
   - аналіз середовища розробки, мов програмування та засобів розробки додатку;
   - особливості реалізації програми;
   - аналіз Android-додатку з додавання та пошуку заходів.

5. Перелік графічного матеріалу
   - структурна схема розроблюваного додатку.
   - алгоритм взаємодії користувача з програмою;
   - діаграма класів програмного продукту;

- презентація за тематикою роботи.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент		

7. Дата видачі завдання:

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою проєкту	18.11.2020	
2.	Розробка та узгодження технічного завдання	30.11.2020	
3.	Аналіз існуючих додатків	10.01.2021	
4.	Підготовка матеріалів для першого розділу проєкту	18.01.2021	
5.	Підготовка матеріалів для другого розділу проєкту	16.02.2021	
6.	Підготовка матеріалів для третього розділу проєкту	14.03.2021	
7.	Створення графічної частини	01.04.2021	
8.	Оформлення документації для дипломного проєкту	20.04.2021	
9.	Попередній огляд матеріалів дипломного проєкту	02.05.2021	

Студент _____ Тимур ПАНКОВ

Керівник проєкту _____ Катерина ПОТАПОВА

*Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (50 с., 27 рис., 4 додатки).

Ключові слова: АНДРОЇД, ДОДАТОК, ПОШУК ЗАХОДІВ, ДОДАВАННЯ ЗАХОДІВ, БАЗА ДАНИХ, EVENTS, DATABASE, JAVA, ANDROID, ANDROID STUDIO, SQLite, FIREBASE.

Об’єкт розробки – Android-додаток з власною базою даних SQLite та сервісами Google (такі як Google Maps) для пошуку та додавання заходів.

Для постановки завдання було зроблено детальний аналіз існуючих додатків зі схожою тематикою та виявлено переваги та недоліки. З урахуванням усіх даних було визначено потребу в розробці додатку з такими характеристиками:

- пошук та додавання заходів;
- створення власної бази даних SQLite всередині програмного продукту;
- відображення на картах заходів;
- доступність перегляду в режимі офлайн;
- відсутність реклами в додатку;
- зручний інтерфейс користувача, який не перевантажений надмірною кількістю непотрібної інформації.

Для розробки додатку було обрано середовище розробки Android Studio від компанії JetBrains, створене спеціально для програмування додатків на базі ОС Android. Мовою програмування обрано Java через її зручність та розповсюдженість, що дозволяє вирішити багато проблем, виявлених в ході аналізу, та підключити велику кількість додаткових сервісів, що не є влаштованими за замовчуванням. У програму внесена власна база даних SQLite, адже вона має влаштовану підтримку Android Studio та є найбільш пристосованою для обраного середовища розробки.

В ході виконання дипломного проєкту:

- проведено аналіз операційної системи Android;

- виконано огляд існуючих рішень проблеми;
- обрано та аргументовано засоби розробки додатку;
- визначено архітектуру системи;
- описано компоненти програми;
- створено Android-додаток з урахуванням переваг та недоліків існуючих аналогів.

Впровадження цього додатку дозволить користувачу швидко та зручно знаходити заходи та додавати власні у зручному інтерфейсі на власному гаджеті на базі ОС Android. Розробникам додаток дозволить використовувати створений продукт, як приклад використання власної бази даних, мап Google в проєкті.

ANNOTATION

Qualification work includes an explanatory note (50 p., 27 fig., 4 appendices).

Keywords: ANDROID, APPLICATION, ACTIVITY SEARCH, ACTIVITY ADD, DATABASE, EVENTS, DATABASE, JAVA, ANDROID, ANDROID STUDIO, SQLite, FIREBASE.

The object of development is an Android application with its own SQLite database and Google services (such as Google Maps) for finding and adding events.

To set the task, a detailed analysis of existing applications with similar topics was identified and the advantages and disadvantages were identified. Taking into account all the data, the need to develop an application with the following characteristics was determined:

- search and add events;
- creating your own SQLite database within the software product;
- display on action maps;
- availability of offline viewing;
- no advertising in the application;
- user-friendly interface that is not overloaded with an excessive amount of unnecessary information.

To develop the application, we chose the Android Studio development environment from JetBrains, created specifically for programming applications based on the Android OS. The programming language is Java because of its convenience and prevalence, which allows you to solve many problems identified during the analysis, and connect a large number of additional services that are not arranged by default. The program includes its own SQLite database, as it has Android Studio support and is best suited for the chosen development environment.

During the implementation of the diploma project:

- analysis of the Android operating system;

- review of existing solutions to the problem;
- selected and argued tools for application development;
- the architecture of the system is defined;
- program components are described;
- An Android application was created taking into account the advantages and disadvantages of existing analogues.

The implementation of this application will allow the user to quickly and easily find events and add their own in a user-friendly interface on their own gadget based on Android. The application will allow developers to use the created product as an example of using their own database, Google Maps in the project.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.002 ТЗ	Events-додаток мовою	4		
			програмування Java			
			Технічне завдання			
	А4	ІАЛЦ.467200.003 ТП	Events-додаток мовою	2		
			програмування Java			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.467200.004 ПЗ	Events-додаток мовою	59		
			програмування Java			
			Пояснювальна записка			
	А4	ІАЛЦ.467200.005 Д1	Events-додаток мовою	1		
			програмування Java			
			Діаграма класів.			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.001 ОА		
Розробив		Панков Т.С.			Events-додаток мовою програмування Java **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Потапова К.Р.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-72	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.006 Д2	Events-додаток мовою	1		
			програмування Java.			
			Взаємодія класів.			
			Схема структурна			
	А4	ІАЛЦ.467200.007 Д3	Events-додаток мовою	1		
			програмування Java.			
			Додавання заходу.			
			Схема алгоритму			
	А4	ІАЛЦ.467200.008 Д4	Events-додаток мовою	1		
			програмування Java			
			Пошук шляху до заходу.			
			Схема алгоритму			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал.			
			Презентація проєкту			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.001 ОА	Арк. 2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Events-додаток мовою програмування Java».

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення мобільного додатку під операційну систему Android для додавання та пошуку заходів з власною базою даних та підключенням до мап Google.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- сумісність з операційною системою Android;
- можливість додавання та пошуку заходів;
- наявність влаштованої бази даних SQLite;
- підключення до сервісу Google Maps.

5.2 Вимоги до апаратного забезпечення

- 8 GB RAM;
- наявність SSD пам’яті;
- Роздільність дисплею 1440x900.
- Тож потрібно зауважити реальні системні вимоги:
- Microsoft Windows 10/8/7 (32- or 64-bit), Mac OS X 10.10 або вище, графічне середовище GNOME чи KDE для ОС Linux;
- 8 GB RAM мінімум;
- мінімум 12 GB простору;
- SSD пам’ять;
- Java Development Kit (JDK) 8;
- Роздільність дисплею 1440x900 мінімум.

5.3 Вимоги до програмного та апаратного забезпечення користувача

- операційна система Android 5.0 або вище.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Видача завдання на дипломне проєктування	04.11.2020
2.	Вивчення літератури за тематикою роботи	16.11.2020
3.	Розроблення та узгодження технічного завдання	25.11.2020
4.	Розроблення структури додатку	17.01.2021
5.	Розроблення дизайну та графічних елементів	04.02.2021
6.	Програмна реалізація додатку	15.03.2021
7.	Тестування додатку	04.04.2021
8.	Підготовка матеріалів текстової частини проєкту	24.04.2021
9.	Підготовка матеріалів графічної частини проєкту	17.05.2021
10.	Оформлення технічної документації проєкту	28.05.2021

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

- ІАЛЦ.467200.005 Д1. Events-додаток мовою програмування Java. Діаграма класів. Схема структурна
- ІАЛЦ.467200.006 Д2. Events-додаток мовою програмування Java. Взаємодія класів. Схема структурна
- ІАЛЦ.467200.007 Д3. Events-додаток мовою програмування Java. Додавання заходу. Схема алгоритму
- ІАЛЦ.467200.008 Д4. Events-додаток мовою програмування Java. Пошук шляху до заходу. Схема алгоритму

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ОС – операційна система;

ПЗ – Програмне забезпечення;

ANDROID – операційна система і платформа мобільних телефонів та планшетних комп'ютерів на базі ядра Linux;

API – Application Programming Interface – прикладний програмний інтерфейс;

Linux – Unix-подібна операційна система;

Dalvik – заснована на регістрах віртуальна машина, розроблена як частина мобільної платформи Android;

FIREBASE – серверна база даних та сервер для роботи додатку;

GNU – вільна UNIX-подібна операційна система;

Gradle – система автоматичного збирання, яка використовує предметно-оріентовану мову (DSL) замість традиційної XML-подібної форми представлення конфігурації проєкту;

IDE – інтегроване середовище розробки;

IEEE – Institute of Electrical and Electronics Engineers – Інститут Інженерів з Електротехніки та Електроніки;

Java – мова програмування, яка використовується в даному проєкті для написання додатку на базі Android;

JDK – Java Development Kit;

JetBrains – російська компанія з розробки програмного забезпечення;

JVM – Java Virtual Machine;

MVC – шаблон архітектури ПЗ Model-View-Controller;

MVP – шаблон архітектури ПЗ Model-View-Presenter;

MVVM – шаблон архітектури ПЗ Model-View-ViewModel;

OpenGL – Open Graphics Library – відкрита графічна бібліотека;

ProGuard – утиліта командного рядка, призначена для оптимізації Java коду;

RAM – пам’ять з довільним доступом;

SDK – Software development kit – набір із засобів розробки, утиліт і документації для створення прикладних програм;

SSD – solid-state drive – твердотілий накопичувач;

SQLite – полегшена реляційна система керування базами даних;

WPF – Windows Presentation Foundation – графічна презентаційна підсистема;

XML – Extensible Markup Language – стандарт побудови структурованих даних для обміну між різними застосунками.

ВСТУП

Мобільні пристрої в наш час невід'ємна складова життя сучасної людини. Адже це комунікація між людьми, прослуховування улюбленої музики, перегляд аудіовізуальної інформації, записник та багато іншого.

На сьогоднішній день найбільш розповсюдженою оперативною системою є Android, так як він підтримує велику кількість пристроїв від різних виробників. Головною причиною розповсюдження ОС Android серед розробників є безкоштовні та зрозумілі засоби розробки додатків для цієї платформи, а для користувачів – доступність гаджетів.

При пошуку додатків для завантаження на свій пристрій користувачі обирає найбільш зручні та потрібні для користування програми. Однією з потрібних програм, особливо для молоді, є додатки з пошуку місця для розваг, навчання, знайомств, святкувань має досить великий попит, особливо в період пандемії, яка значно зменшила кількість таких заходів.

Наразі існує достатньо велика кількість додатків з пошуку розваг, навчальних заходів, тощо, проте більшість з них зосереджені на досить популярні, відомі та вже відвідувані заходи або мають дуже вузьку спеціалізацію, також більшість не має функції додавання власних подій. З метою прибутку, розробники додають надмірну кількість реклами у свої додатки, що робить незручним інтерфейс користувача. Більшість додатків не мають або мають незручну функцію геолокації та пошуку шляху до обраного заходу. Через використання баз даних, що зберігаються в мережі інтернет, роблять неможливим використання додатків у режимі офлайн. Особливо важливим є факт, що на базі iOS таких додатків у рази більше, ніж на базі ОС Android, що зменшує вибір бажаного додатку.

Можна зауважити, що досить непогані додатки, що вирішують дану проблему, зосереджені в країнах західної Європи та США, що наголошує на актуальності проєкту в Україні.

Дивлячись на все це можна зауважити, що створення додатка на базі ОС Android з можливістю додавання власних та пошуку існуючих даних зі зручним інтерфейсом користувача, який не містить реклами, має власну базу даних заходів, з функціями геолокації, геокодування та влаштованим пошуком шляху до обраного місця є важливою розробкою.

Розроблений мобільний додаток буде корисним також і для розробників, які зможуть переглянути зручність використання власної бази даних додатку та підключення мап Google у додаток.

1. АНАЛІЗ ДОДАТКІВ З ПОШУКУ ЗАХОДІВ НА БАЗІ ОС ANDROID ТА ОБҐРУНТУВАННЯ ТЕМИ

1.1 Особливості та характеристики ОС Android

Android - це мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, призначена в основному для сенсорних мобільних пристроїв, таких як смартфони та планшети. Android розробляється консорціумом розробників, відомим як Open Handset Alliance, і комерційно фінансується Google.

Це безкоштовне програмне забезпечення з відкритим кодом; його вихідний код відомий як Android Open Source Project (AOSP), який в основному ліцензується за ліцензією Apache. Однак більшість пристроїв Android постачаються з попередньо встановленим власним програмним забезпеченням, зокрема Google Mobile Services (GMS), яке включає такі основні програми, як Google Chrome, платформа цифрового розповсюдження Google Play та пов'язана з ними платформа розробки служб Google Play. Близько 70 відсотків смартфонів Android управляють екосистемою Google; конкуруючі екосистеми та форки Android включають Fire OS (розроблена Amazon) або LineageOS. Однак назва та логотип "Android" є товарними знаками Google, які встановлюють стандарти щодо обмеження використання "несертифікованих" пристроїв за межами їх екосистеми для використання бренду Android.

Вихідний код був використаний для розробки варіантів Android для цілого ряду іншої електроніки, таких як ігрові консолі, цифрові камери, портативні медіаплеєри, ПК та інші, кожен зі спеціалізованим інтерфейсом користувача. Деякі відомі похідні версії включають Android TV для телевізорів та Wear OS для переносних пристроїв, розроблені Google. Пакети програмного забезпечення для Android, які використовують формат APK, зазвичай розповсюджуються через власні магазини додатків, такі як Google Play Store, Samsung Galaxy Store, Huawei

AppGallery, Cafe Bazaar та GetJar, або платформи з відкритим кодом, такі як Aptoide або F-Droid.

Android є найбільш продаваною ОС у світі на смартфонах з 2011 року, а на планшетах - з 2013 року. Станом на травень 2017 року вона має понад два мільярди активних користувачів щомісяця, найбільшу встановлену базу будь-якої операційної системи, а станом на січень 2021 року Google Play Store має понад 3 мільйони програм. Поточна стабільна версія - Android 11, випущена 8 вересня 2020 року.

Базовим елементом операційної системи є реалізація Dalvik віртуальної машини Java і все програмне забезпечення та застосування спираються на цю реалізацію Java.

Кодове ім'я кожного великого релізу Android, починаючи з версії 1.5, являє собою назву якого-небудь десерту. Перші букви найменувань в порядку версій відповідають літерами латинського алфавіту:

- 1.5 Cupcake («*кекс*»),
- 1.6 Donut («*пончик*»),
- 2.0/2.1 Eclair («*еклер*» або «*глазур*»),
- 2.2 Froyo (скорочення від «*заморожений йогурт*»),
- 2.3 Gingerbread («*імбирний пряник*»),
- 3.0 Honeycomb («*медові стільники*»),
- 4.0 Ice Cream Sandwich («*брикет морозива*»),
- 4.1/4.2/4.3 Jelly Bean («*желейні боби*»),
- 4.4 KitKat (на честь однойменного бренду шоколадних батончиків «*КітКат*»);
- 5.0/5.1 Lollipop («*льодяник*»),
- 6.0 Marshmallow («*зефір*»),
- 7.0/7.1 Nougat — Nougat («*нуга*»);

- 8.0/8.1 Oreo — Oreo (печиво «*Орео*»);
- 9.0 Pie — Pie («*пиріг*»);
- 10 Q — Q (назва не розшифровується);

Багато власників гаджетів на базі ОС Android не завжди хочуть оновлювати операційну систему до найвищої через велику кількість причин, наприклад, зручність та звичність користування старою версією, через це і розповсюдженість версій дуже велика (рисунок 1.2).

Рисунок 1.1 – Оновлення версій Android

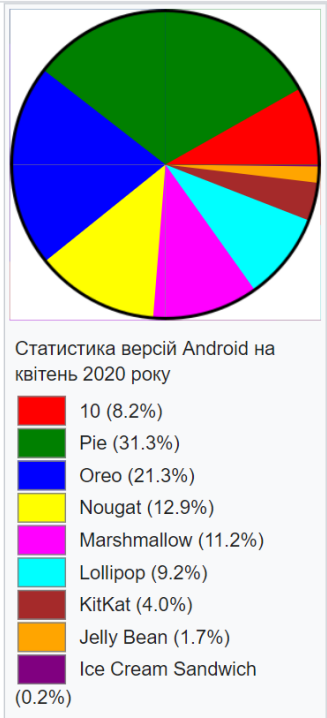


Рисунок 1.2 – Статистика версій Android (квітень 2020р.)

Користувальницький інтерфейс Android за замовчуванням в основному заснований на прямих маніпуляціях, використовуючи сенсорні входи, які вільно

відповідають дійсним діям, таким як проведення, натискання, стискання та зворотне стискання для маніпулювання об’єктами на екрані, разом з віртуальною клавіатурою. Ігрові контролери та повнорозмірні фізичні клавіатури підтримуються через Bluetooth або USB. Внутрішнє обладнання, таке як акселерометри, гіроскопи та датчики наближення, використовується деякими програмами для реагування на додаткові дії користувача, наприклад, регулювання екрану від книжкової до альбомної залежно від того, як орієнтований пристрій.

Пристрої Android завантажуються на головний екран, основний навігаційно-інформаційний «хаб» на пристроях Android, аналогічно робочому столу, що знаходиться на персональних комп’ютерах. Домашні екрани Android зазвичай складаються з піктограм додатків та віджетів; піктограми програм запускають відповідну програму, тоді як віджети відображають оновлюваний вміст, який автоматично оновлюється, наприклад прогноз погоди, поштову скриньку користувача чи індикатор новин безпосередньо на головному екрані.

У верхній частині екрана знаходиться рядок стану, що відображає інформацію про пристрій та його зв’язок. Цей рядок стану можна потягнути (проведіть) вниз, щоб відкрити екран сповіщень, де програми відображають важливу інформацію або оновлення, а також швидкий доступ до системних елементів керування та перемикань, таких як яскравість дисплея, налаштування підключення (WiFi, Bluetooth, стільникові дані), аудіорежим та ліхтарик.

Сповіщення - це "коротка, своєчасна та відповідна інформація про вашу програму, коли вона не використовується", а при натисканні користувачі переходять на екран усередині програми, що стосується сповіщення.

На екрані "Усі програми" перераховані всі встановлені програми з можливістю перетягування програми зі списку на головний екран. Екран «нещодавно відкриті» дозволяє користувачам переключатися між нещодавно використовуваними програмами.

Багато ранніх смартфонів з ОС Android були обладнані спеціальною кнопкою пошуку для швидкого доступу до веб-пошукової системи та функцією внутрішнього пошуку окремих програм.

Майже всі пристрої Android, мають попередньо встановлені програми Google, включаючи Gmail, Карти Google, Google Chrome, YouTube, Google Play Музику, Google Play Фільми та ТБ та багато іншого.

Додатки, які розширюють функціональність пристроїв (і повинні бути 64-розрядними), створюються за допомогою набору для розробки програмного забезпечення Android (SDK) і мови програмування. Java та/або інші мови JVM, такі як Kotlin, можуть поєднуватися з C / C ++, разом з вибором стандартних середовищ виконання, які забезпечують кращу підтримку C ++. Мова програмування Go також підтримується, хоча з обмеженим набором інтерфейсів прикладного програмування (API).

Android має все більший вибір сторонніх додатків, які користувачі можуть придбати, завантаживши та встановивши файл APK програми (пакет додатків Android), або завантаживши їх за допомогою програми магазину програм, яка дозволяє користувачам встановлювати, оновлювати та видаляти програми зі своїх пристроїв. Google Play Store - це основний магазин додатків, встановлений на пристроях Android, які відповідають вимогам сумісності Google і ліцензують програмне забезпечення Google Mobile Services.

Оскільки пристрої Android, як правило, живляться від акумулятора, Android призначений для управління процесами, щоб звести енергію до мінімуму. Коли програма не використовується, система призупиняє свою роботу, так що, незважаючи на те, що вона доступна для негайного використання, а не закрита, вона не використовує заряд акумулятора або ресурси центрального процесора. Android автоматично управляє програмами, що зберігаються в пам'яті: коли пам’яті мало, система почне непомітно і автоматично закриватиме неактивні процеси, починаючи з тих, які були неактивними найдовше.

Основною апаратною платформою для Android є ARM (архітектури ARMv7 та ARMv8-A), архітектури x86 та x86-64 також офіційно підтримуються в пізніших версіях Android. Неофіційний проєкт Android-x86 забезпечив підтримку архітектур x86 напередодні офіційної підтримки.

З 2012 року почали з'являтися пристрої Android з процесорами Intel, включаючи телефони та планшети. Отримавши підтримку 64-розрядних платформ, Android спочатку працював на 64-розрядному x86, а потім на ARM64. Оскільки Android 5.0 "Lollipop", на додаток до 32-розрядних варіантів підтримуються 64-розрядні варіанти всіх платформ.

Платформа легко пристосовується для використання VGA, бібліотек двовимірної і тривимірної графіки, розроблених на основі OpenGL ES 1.0-3.1 специфікації, традиційних інструментаріїв для смартфонів.

ОС Android використовує бази даних SQLite для структурованих даних.

Програми, написані на Java, можна скомпілювати в Dalvik байткод і виконувати на Dalvik virtual machine, яка являє собою розроблену спеціально для використання на мобільних пристроях віртуальну машину, незважаючи на те, що не є стандартною JVM.

Переваги ОС Android:

- велика різноманітність додатків;
- швидка інтеграція з сервісами Google;
- наявність файлової системи;
- незалежність від апаратного «заліза» мобільного пристрою;
- є системою з відкритим кодом, що призвело до появи величезної кількості корисних додатків, більшість з яких можна скачати абсолютно безкоштовно, крім того, в телефоні при покупці вже будуть стояти найпопулярніші і необхідні додатки, такі як Google+, Gmail, Google Maps;
- багатозадачність;
- легкість інсталювання додатків з різних ресурсів;

- широкі можливості індивідуалізації;
- підтримка флеш-програвача;
- зручне, своєчасне та швидке оновлення системи;
- можливість заміни/видалення стандартного «заліза».
- спрощене виробництво додатків, ігор, плагінів, оновлень;
- при підключенні телефону до комп'ютера через USB кабель пристрій розпізнається як знімний накопичувач, в результаті на телефон і карту пам'яті можна скопіювати будь-яку інформацію звичним способом.

1.2 Огляд існуючих додатків з пошуку заходів

Після аналізу існуючих додатків було визначено ряд недоліків та переваг, що допомогло звернути увагу на нагальні проблеми для подальшого їх вирішення.

Одним з популярних додатків з пошуку заходів можна виділити – «AroundMe».

Функціонально додаток дозволяє шукати місця (банкомати (рисунок 1.3), ресторани, кінотеатри та інші) поблизу вас, але з ввімкненням функції геолокації на телефоні користувача, проте адреса не відображається коректно, при ручному вводі всі адреси відображаються в хаотичному порядку. Також існують невеликі незручності при користуванні мапою, такі як неможливість пересунути мапу або прокласти шлях без відкриття додаткових програм (рисунок 1.4). При пошуку цікавлячих заходів відображується надмірна кількість реклами, що заважає користувачу.

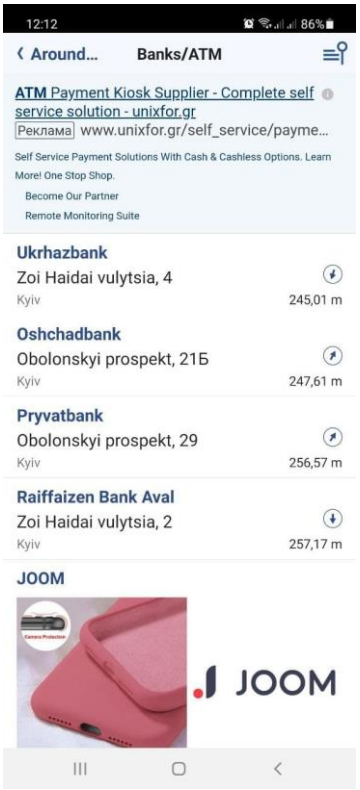
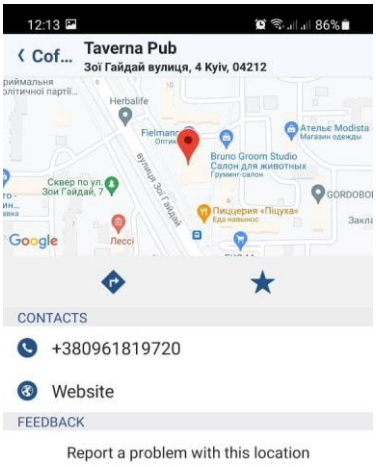


Рисунок 1.3 - Пошук банкоматів в додатку AroundMe



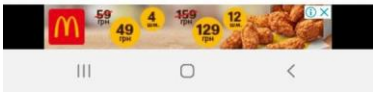


Рисунок 1.4 - Відображення мапи в додатку AroundMe

Також одним з найбільш завантажуваних додатків для пошуку є «Eventbrite».

Додаток більш зосереджений на країни ЄС та США, проте має застосування і в Україні. Для нашої країни в додатку представлені по більшій частині онлайн заходи зарубіжних авторів та заходи в містах України, але більшість з яких платна, на момент написання додаток знайшов лише 12 заходів в місті Київ. При відкриванні додатку користувач отримує повідомлення про заходи онлайн та застереження від коронавірусу (рисунок 1.5), проте інформацію про заходи офлайн треба шукати далі. Пошук заходів супроводжується тегами (рисунок 1.6). Для отримання детальнішої інформації та її подальшого зберігання потрібна авторизація через сервіси Google або Facebook, що може бути небезпечним через витік особистої інформації користувачів.

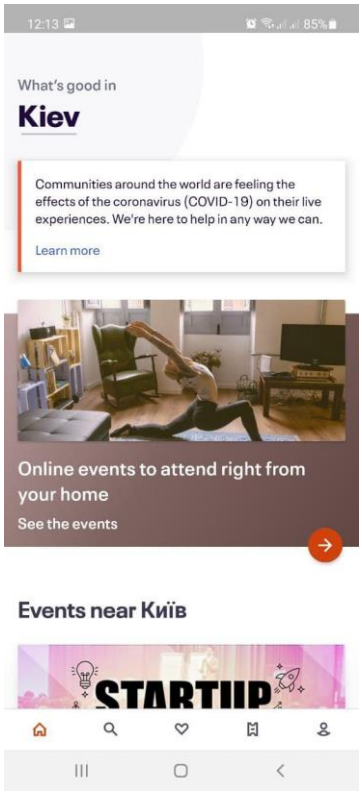


Рисунок 1.5 - Стартовий екран Eventbrite

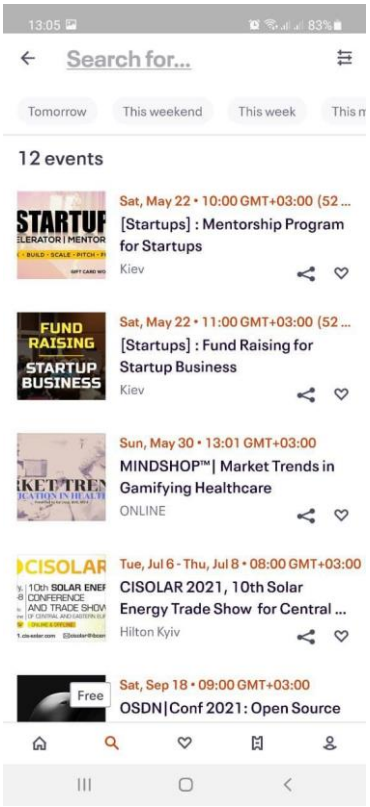


Рис. 1.6 - Пошук заходів в додатку Eventbrite

Отже, опрацювавши існуючі додатки, можна зробити висновок, що робота є необхідною та має містити зручний інтерфейс користувача, пошук існуючих та додавання власних заходів, зручний пошук на мапі для чого потрібні геолокація та геокодування, також містити власну базу даних для зручної роботи додатку в офлайн режимі.

1.3 Постановка та формулювання завдання розробки додатку

Взявши до уваги усі переваги та недоліки існуючих популярних додатків, можна уявити собі «ідеальний» додаток з пошуку та додавання заходів. Даний додаток має містити в собі власну базу даних заходів, в якій буде міститись детальна інформація про організатора, спосіб зв’язку з ним, дату, час, деталі заходу та місце проведення.

Також велику роль відіграє функція пошуку місця на мапі та прокладання маршруту до нього. Такий додаток повинен мати влаштований сервіс Google

Maps, адже така функція допоможе зручно спланувати маршрут для відвідування того чи іншого заходу.

Ще одна важлива деталь – це інтерфейс користувача. Він має бути привітний, зручний та зрозумілий користувачу. Так як інтерфейс додатку – це інструмент, з допомогою якого ведеться взаємодія користувача з програмою, саме він складає перше враження від додатку.

Важлива деталь – функція пошуку потрібного заходу в базі даних. Доступ має надаватися в режимі офлайн, так як не завжди є доступ до мережі Інтернет.

Виходячи з усього перерахованого, основними функціями в додатку мають бути:

- зручний інтерфейс користувача;
- наявність власної бази даних;
- функції геолокації та геокодування;
- функція пошуку;
- робота додатку в режимі офлайн.

Багато існуючих додатків орієнтовані на відомі та розкручені заходи, які легко знайти і в Інтернеті, і в месенджерах, і в спеціальних пошукових додатках. З чого можна зробити висновок, що написання додатку для менш популярних заходів та організаторів є дуже важливим проєктом.

2. ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ

2.1 Опис концепції розроблюваного програмного забезпечення

Для розробки додатків під ОС Android використовують багато мов програмування, такі як Java, Kotlin, C++, Python та інші.

Згідно рейтингу IEEE (рисунок 2.1), який включає в себе мови, які використовуються на GitHub, визначив найбільш популярні та зручні мови програмування для Android розробок. Вибірка була відносно частоти зустрічі мови на 8 різних майданчиках і сайтах, наприклад, StackOverflow. Google Trends та Reddit. Також до мов прив'язані теги сфери застосування.

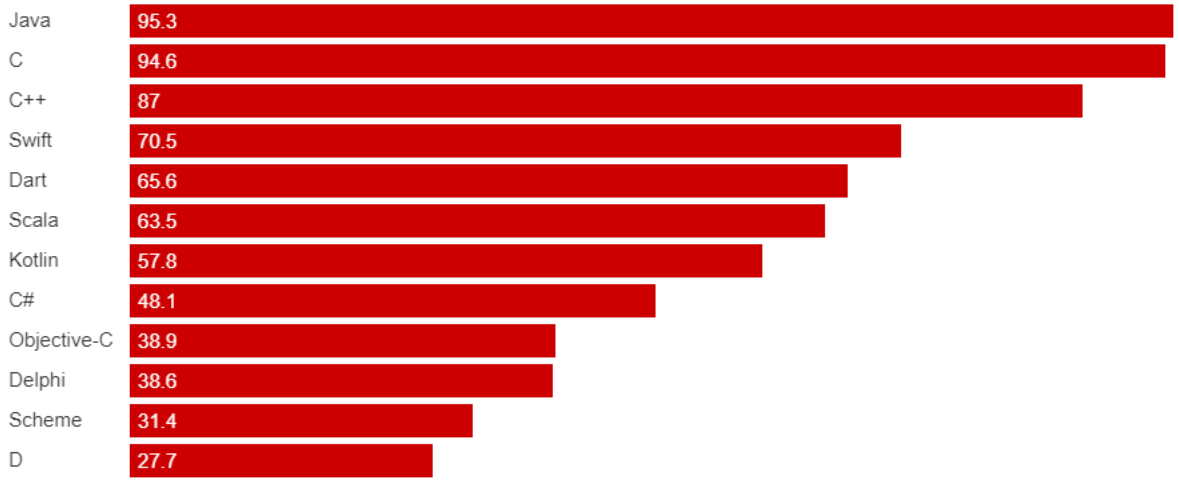


Рисунок 2.1 – Рейтинг IEEE

Лідером в області мобільної розробки є мова програмування Java. Проте компанія Google продовжує нав'язувати розробникам мову Kotlin, як основну для Android-розробки. Саме тому ця молода мова має досить великий рейтинг. Не зважаючи на це експерти впевнені, що Java буде домінувати ще великий проміжок часу на ринку розробки додатків до операційної системи Android.

Рекомендованою середою розробки є Android Studio, адже вона створена спеціально для розробки додатків під ОС Android. Android Studio - це офіційне інтегроване середовище розробки (IDE) для розробки додатків для Android, засноване на IntelliJ IDEA. На додаток до потужного редактора коду та інструментів розробника IntelliJ, Android Studio пропонує ще більше функцій, що

підвищують продуктивність розробника при створенні програм для Android, таких як:

- гнучка система побудови на основі Gradle;
- швидкий і багатофункціональний емулятор;
- єдине середовище, де користувачі можуть розробляти всі пристрої Android;
- присутня можливість застосування змін для надсилання змін коду та ресурсів до запущеної програми без перезапуску програми;
- шаблони коду та інтеграція GitHub, щоб допомогти розробникам створити загальні функції програми та імпортувати зразок коду;
- широкі інструменти та основи тестування;
- інструменти Lint для виявлення продуктивності, зручності використання, сумісності версій та інших проблем;
- підтримка C ++ та NDK;
- вбудована підтримка Google Cloud Platform, що спрощує інтеграцію Google Cloud Messaging та App Engine.

Кожен проєкт в Android Studio містить один або кілька модулів із файлами вихідного коду та файлами ресурсів. Типи модулів включають:

- модулі додатків для Android;
- бібліотечні модулі;
- модулі Google App Engine;

За замовчуванням Android Studio відображає файли проєкту у поданні проєкту Android (рисунок 2.2). Цей вигляд організований за допомогою модулів, щоб забезпечити швидкий доступ до ключових вихідних файлів проєкту.

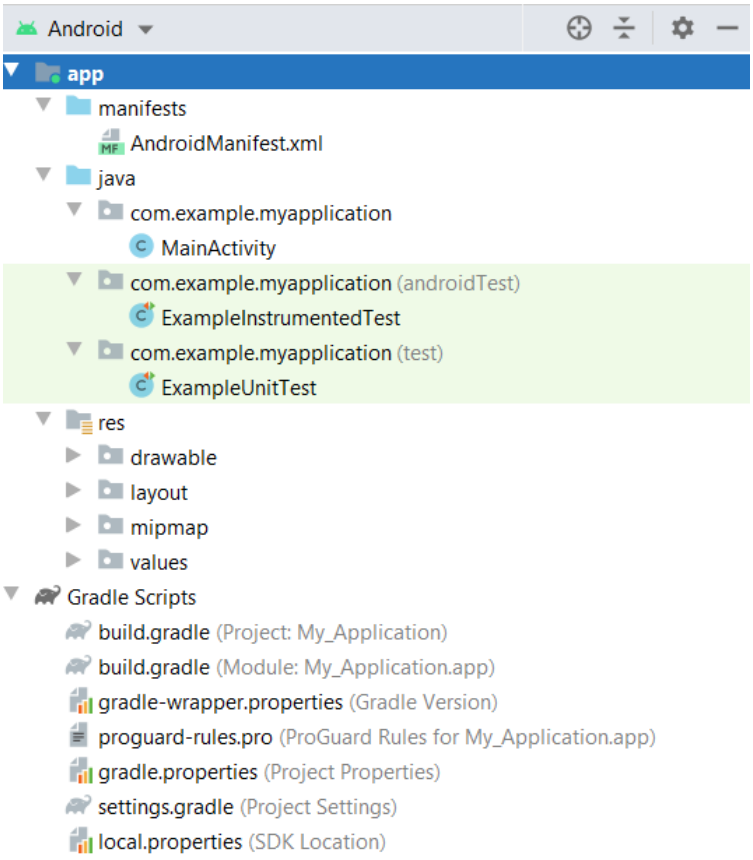


Рисунок 2.2 – Подання проєкту Android

Усі файли збірки видно на верхньому рівні під сценаріями Gradle, і кожен модуль програми містить наступні папки:

- manifests: містить файл AndroidManifest.xml;
- java: містить файли вихідного коду Java, включаючи тестовий код ExampleUnitTest;
- res: містить усі некодові ресурси, такі як макети XML, рядки інтерфейсу користувача та растрові зображення.

Android Studio використовує Gradle як основу систему збірки, з додатковими функціями Android, які надає плагін Android для Gradle. Ця система побудови працює як інтегрований інструмент із меню Android Studio та незалежно від командного рядка. Розробники можуть використовувати функції збірки, щоб зробити наступне:

- налаштувати та продовжити процес збірки;

- створити декілька файлів .apk для свого додатка з різними функціями за допомогою одного проекту та модулів;
- повторно використати код та ресурси у наборах джерел.

Застосовуючи гнучкість Gradle, можливо досягти всього цього, не змінюючи основні вихідні файли програми. Файли збірки Android Studio називаються build.gradle. Це текстові файли, які використовують синтаксис Groovy для налаштування збірки з елементами, наданими плагіном Android для Gradle. Кожен проект має один файл збірки верхнього рівня для всього проекту та окремі файли збірки рівня модуля для кожного модуля. Під час імпорту існуючого проекту Android Studio автоматично генерує необхідні файли збірки.

Система збірки може допомогти створити різні версії одного додатка з одного проекту. Це корисно, коли є як безкоштовна версія, так і платна версія вашого додатка, або якщо ви хочете розповсюдити кілька файлів .apk для різних конфігурацій пристроїв у Google Play.

У середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проєктування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

Залежності для проекту вказані за іменем у файлі build.gradle. Gradle піклується про пошук залежностей та надання їх доступності у збірці. Існує можливість оголосити залежності модулів, віддалені двійкові залежності та локальні двійкові залежності у файлі build.gradle. Android Studio налаштовує проєкти на використання центрального сховища Maven за замовчуванням.

Використання вбудованих налагоджень, щоб покращити проходження

коду у поданні налагоджувача за допомогою вбудованої перевірки посилань, виразів та значень змінних. Інформація про вбудовані налагодження включає:

- вбудовані значення змінних;
- посилання на об'єкти, які посилаються на вибраний об'єкт;
- метод повертає значення;
- лямбда та операторні вирази;
- значення підказки.

Android Studio пропонує профайлери продуктивності, завдяки яким можливо легше відстежувати пам’ять програми та використання центрального процесора, знаходити вивільнені об’єкти, знаходити витоки пам’яті, оптимізувати графічну продуктивність та аналізувати мережеві запити. Коли додаток працює на пристрої або емуляторі, треба відкрити вкладку Android Profiler.

Профілюючи використання пам’яті в Android Studio, існує можливість одночасно ініціювати збір сміття та скинути купу Java у знімок купи у файлі двійкового формату HPROF, специфічному для Android. Засіб перегляду HPROF відображає класи, екземпляри кожного класу та дерево посилань, щоб допомогти відстежувати використання пам'яті та знаходити витоки пам'яті.

Використання Memory Profiler присутнє для відстеження виділення пам’яті та спостереження за тим, де виділяються об’єкти під час виконання певних дій. Знання цих розподілів дозволяє оптимізувати продуктивність програми та використання пам’яті, коригуючи виклики методів, пов’язані з цими діями.

Google пропонує для вільного завантаження інструментарій для розробки (SDK), який призначений для x86-машин під операційними системами Linux, Mac OS X (10.4.8 або вище), Windows 7 та вище. Для розробки потрібен Java Development Kit 5 або новіший.

Програмні методи і бібліотеки змінюються. Якісь стають застарілими і їх потрібно заміняти на більш нові. Таким чином стає вибір – підтримувати новіші

функції ОС або дозволити більшій кількості користувачів інсталювати додаток.

В останніх версіях ОС додали багатозадачність робочої області. Користувач може відобразити на робочій поверхні одночасно декілька додатків, що може призвести до виділення довільної частини екрану для додатку (менше або більше, ніж потрібно).

Тестування проєкту має відбуватися на декількох різних фізичних пристроях. Адже велика кількість смартфонів, планшетів та інших гаджетів використовують різні розміри і екрану, і емблеми, і зображень, які є в додатку.

2.2 Системні вимоги до розробки додатків на ОС Android.

Перед початком розробки додатку для Android потрібно інсталювати обрану середу розробки та всі технічні засоби. При написанні додатку «Events» були інстальовані Android Studio від JetBrains, Java Development Kit 16.0.1 (остання версія) та використовувалась операційна система Windows 10.

В офіційній документації Android Studio вказані такі системні вимоги:

- Microsoft Windows 10/8/7 (32- or 64-bit), Mac OS X 10.10 або вище, графічне середовище GNOME чи KDE для ОС Linux;
- 4 GB RAM мінімум, 8 GB RAM рекомендовано;
- 500 MB простору для Android Studio, щонайменше 4 GB для SDK, зображень емульованих систем та кешів;
- Java Development Kit (JDK) 8;
- Роздільність дисплею 1280x800 мінімум.

При користуванні Android Studio було зауважено реальні системні вимоги:

- 8 GB RAM;
- наявність SSD пам’яті;
- Роздільність дисплею 1440x900.
- Тож потрібно зауважити реальні системні вимоги:
- Microsoft Windows 10/8/7 (32- or 64-bit), Mac OS X 10.10 або вище, графічне середовище GNOME чи KDE для ОС Linux;

- 8 GB RAM мінімум;
- мінімум 12 GB простору;
- SSD пам’ять;
- Java Development Kit (JDK) 8;
- Роздільність дисплею 1440x900 мінімум.

2.3 Засоби розробки додатку

2.3.1 Мова програмування Java

Java - це високорівнева, об'єктно-орієнтована мова програмування на основі класів, яка розроблена з якомога меншою залежністю реалізації. Це мова програмування загального призначення, призначена для того, щоб розробники програм могли писати один раз, а запускати їх де завгодно (WORA – Write Once, Run Anywhere), що означає, що скомпільований код Java може працювати на всіх платформах, що підтримують Java, без необхідності перекомпіляції. Програми Java зазвичай компілюються в байт-код, який може працювати на будь-якій віртуальній машині Java, незалежно від базової архітектури комп'ютера. Синтаксис Java подібний до C та C ++, але має менше засобів низького рівня, ніж будь-який з них. Час виконання Java забезпечує динамічні можливості (наприклад, відображення та модифікацію коду середовища виконання), які, як правило, недоступні в традиційних компільованих мовах.

Спочатку Java була розроблена Джеймсом Гослінгом у Sun Microsystems (яка згодом була придбана Oracle) і випущена в 1995 році як основний компонент Java-платформи Sun Microsystems. Оригінальні та довідкові реалізатори Java-компілятори, віртуальні машини та бібліотеки класів були спочатку випущені Sun під власними ліцензіями. Станом на травень 2007 року, згідно з вимогами Процесу спільноти Java, Sun здійснила ліцензію на більшість своїх технологій Java під загальною публічною ліцензією GNU. Oracle пропонує власну віртуальну машину HotSpot Java, однак офіційним посиланням є OpenJDK JVM, яке є безкоштовним програмним забезпеченням з відкритим кодом і використовується

більшістю розробників і є JVM за замовчуванням для майже всіх дистрибутивів Linux.

Станом на березень 2021 року останньою версією є Java 16, з Java 11, яка наразі підтримується довгостроковою підтримкою (LTS), випущена 25 вересня 2018 р. Oracle випустила останнє публічне оновлення із застарілою версією Java 8 LTS у січні 2019 року для комерційного використання, хоча в іншому випадку він все ще підтримуватиме Java 8 із загальнодоступними оновленнями для особистого користування на невизначений час. Інші постачальники почали пропонувати збірки OpenJDK 8 та 11 з низькою вартістю, які все ще отримують оновлення безпеки та інші оновлення.

У створенні мови Java було п’ять основних цілей:

- синтаксис має бути простим, об’єктно-орієнтованим та звичним;
- мова повинна бути надійним і надійним;
- Java має бути нейтральною до архітектури та портативною;
- присутня висока продуктивність;
- мова повинна бути інтерпретованим, різьбовим та динамічним.

Однією з цілей дизайну Java є портативність, що означає, що програми, написані для платформи Java, повинні працювати аналогічно на будь-якій комбінації обладнання та операційної системи з адекватною підтримкою часу роботи. Це досягається компіляцією коду мови Java до проміжного подання, що називається байт-кодом Java, замість безпосередньо до специфічного для архітектури машинного коду. Інструкції байт-коду Java аналогічні машинному коду, але вони призначені для виконання віртуальною машиною, написаною спеціально для апаратного забезпечення хоста. Кінцеві користувачі зазвичай використовують середовище виконання Java, встановлене на їх машині для окремих програм Java або у веб-браузері для аплетів Java.

Стандартні бібліотеки забезпечують загальний спосіб доступу до особливостей хоста, таких як графіка, створення потоків та створення мереж.

Використання універсального байт-коду спрощує перенесення. Однак накладні витрати на інтерпретацію байт-коду в машинних інструкціях робили інтерпретовані програми майже завжди більш повільними, ніж власні виконувані файли. Компілятори, які компілюють байт-коди до машинного коду під час виконання, були введені з ранньої стадії. Сама Java не залежить від платформи і пристосована до конкретної платформи, на якій вона повинна працювати віртуальною машиною Java для неї, яка переводить байт-код Java на машинну мову платформи.

Критика, спрямована на Java, включає реалізацію дженериків, швидкість, обробку непідписаних чисел, реалізацію арифметики з плаваючою крапкою, та історію вразливих місць безпеки у первинній реалізації JVM HotSpot.

Мова Java є ключовою опорою в Android, мобільній операційній системі з відкритим кодом. Хоча Android, побудований на ядрі Linux, написаний переважно на мові C, Android SDK використовує мову Java як основу для програм для Android, але не використовує жодного зі своїх стандартних графічних інтерфейсів, SE, ME або інших встановлених стандартів Java. Мова байт-коду, що підтримується Android SDK, несумісна з байт-кодом Java і працює на власній віртуальній машині, оптимізованій для пристроїв з малою пам’яттю, таких як смартфони та планшетні комп’ютери. Залежно від версії Android, байт-код інтерпретується віртуальною машиною Dalvik або компілюється в рідний код робочою системою Android.

Ключовий компонент життєвого циклу програми є Activity. Схематично взаємозв’язок між усіма викликами можливо показати так, як на рисунку 2.3.

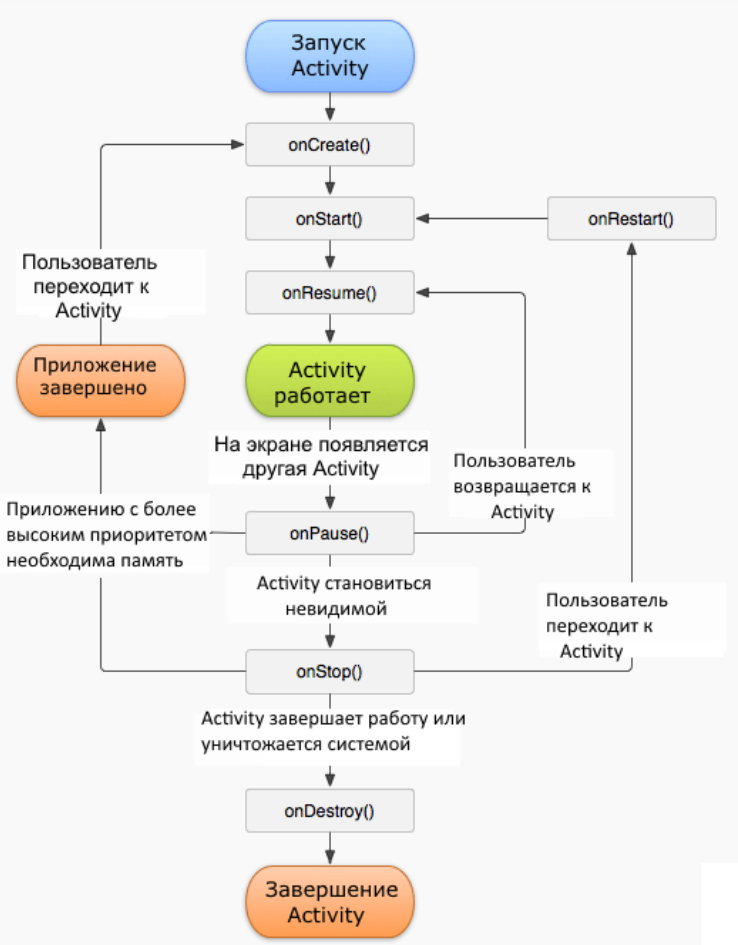


Рисунок 2.3 - Життєвий цикл Activity

Діяльність у системі управляється як стеки дій. Коли починається нова діяльність, вона зазвичай розміщується у верхній частині поточного стеку і стає поточним – попередня діяльність завжди залишається під нею у стеку і знову не вийде на передній план, поки не вийде нова. На екрані може бути видно один або кілька стеків діяльності.

Діяльність має, по суті, чотири стани:

- якщо діяльність знаходиться на передньому плані екрана, то вона активна або запущена, зазвичай це діяльність, з якою користувач зараз взаємодіє;
- якщо дія втратила фокус, але все ще представлена користувачеві, це візуально помітно. Це можливо, якщо нова не повнорозмірна або прозора

діяльність фокусується на вашій діяльності, інша діяльність займає вищу позицію в режимі багато вікон, або сама діяльність не може бути сфокусованою у поточному режимі вікон. Така діяльність повністю жива (вона зберігає всю інформацію про стан та членів і залишається прикріпленою до менеджера вікон);

- якщо діяльність повністю закрита іншою, вона зупиняється або приховується. Він як і раніше зберігає всю інформацію про стан та членів, однак користувачеві більше не видно, тому його вікно приховано, і система часто його знищує, коли потрібна пам’ять в іншому місці;
- система може скинути діяльність з пам'яті, попросивши її закінчити, або просто знищивши її процес, зробивши його знищеним. Коли він знову відображається користувачеві, його слід повністю перезапустити та відновити до попереднього стану.

onCreate() викликається, коли дія створюється вперше. Тут слід виконувати всі свої звичайні статичні налаштування: створювати подання, прив’язувати дані до списків тощо. Цей метод також надає вам Bundle, що містить попередньо заморожений стан діяльності, якщо такий був. Завжди слідує onStart()

onRestart() викликається після припинення діяльності перед її повторним запуском. Завжди слідує onStart().

onStart() викликається, коли діяльність стає видимою для користувача. Слідом за onResume(), якщо діяльність виходить на перший план, або onStop (), якщо вона стає прихованою.

onResume() викликається, коли дія почне взаємодіяти з користувачем. На цей момент ваша активність знаходиться у верхній частині стеку своїх активностей, і в неї вводиться введення користувачем. Завжди слідує onPause().

onPause() викликається, коли дія втрачає стан переднього плану, більше не фокусується або перед переходом у зупинений або знищений стан. Діяльність все ще видна користувачеві, тому рекомендується залишати її візуально активною та продовжувати оновлювати інтерфейс. Впровадження цього методу повинно бути дуже швидким, оскільки наступна дія не буде відновлена, поки цей метод не повернеться. Слідує або onResume(), якщо дія повертається назад, або onStop(), якщо стає невидимою для користувача.

onStop() викликається, коли діяльність більше не відображається користувачеві. Це може статися або через те, що починається нова діяльність зверху, перед цією висувається існуюча, або ця руйнується. Зазвичай це використовується для зупинки анімації та оновлення інтерфейсу тощо. Слідом або onRestart(), якщо ця активність повертається для взаємодії з користувачем, або onDestroy(), якщо ця активність припиняється.

onDestroy() отримується до того, як діяльність буде знищена. Це може статися або через те, що діяльність закінчується, або тому, що система тимчасово знищує цей екземпляр діяльності, щоб заощадити місце.

Якщо конфігурація пристрою (як визначено класом Resources.Configuration) змінюється, то все, що відображає користувальницький інтерфейс, потрібно буде оновити, щоб відповідати цій конфігурації. Оскільки Activity є основним механізмом взаємодії з користувачем, він включає спеціальну підтримку для обробки змін конфігурації.

Якщо не вказано інше, зміна конфігурації (наприклад, зміна орієнтації екрана, мови, пристроїв введення тощо) призведе до знищення поточної активності, проходячи звичайний процес життєвого циклу onPause(), onStop() та onDestroy() відповідно. Якщо діяльність була на передньому плані або видима користувачеві, після того, як у цьому екземплярі буде викликано onDestroy(),

буде створено новий екземпляр активності з будь-яким збереженим InstanceState, попереднім екземпляром, згенерованим з onSaveInstanceState (Bundle).

Це робиться тому, що будь-який ресурс програми, включаючи файли макета, може змінюватися залежно від будь-якого значення конфігурації. Таким чином, єдиним безпечним способом обробки зміни конфігурації є повторне отримання всіх ресурсів, включаючи макети, малюнки та рядки. Оскільки дії вже повинні знати, як зберегти свій стан і відтворити себе з цього стану, це зручний спосіб перезапустити діяльність з новою конфігурацією.

Перехід між станами activity показує рисунок 2.4.

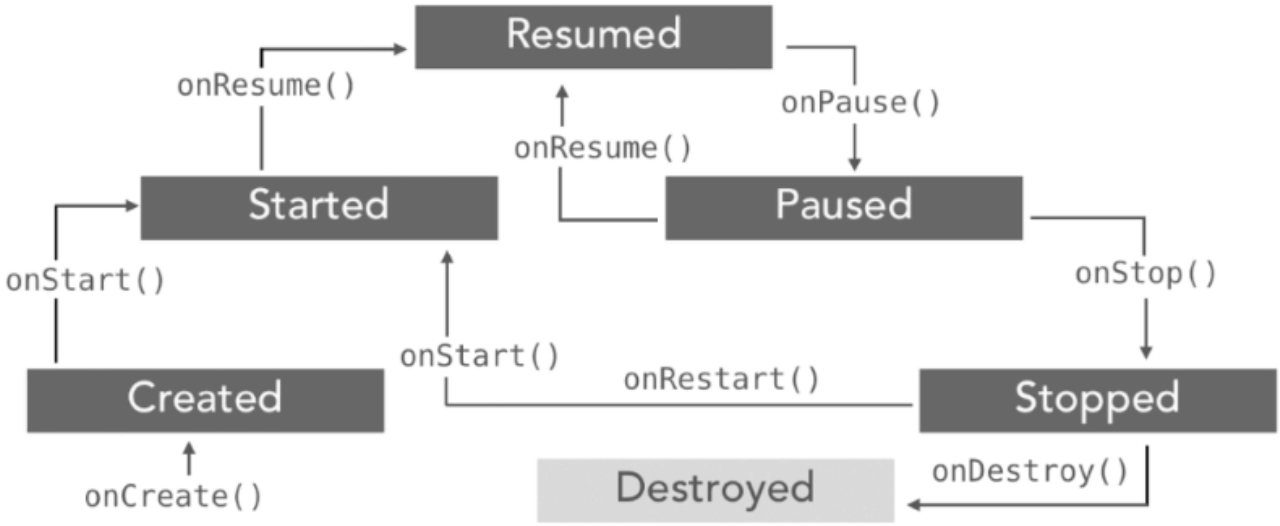


Рисунок 2.4 - Перехід між станами

2.3.2 Ресурси мови XML

Розширювана мова розмітки (XML) - це мова розмітки, яка визначає набір правил для кодування документів у форматі, який читається як людиною, так і машинною. Цілі дизайну XML підкреслюють простоту, загальність та зручність використання в Інтернеті. Це формат текстових даних із потужною підтримкою за допомогою Unicode для різних людських мов. Хоча дизайн XML зосереджений

на документах, мова широко використовується для подання довільних структур даних, таких як ті, що використовуються у веб-сервісах.

Документ XML - це рядок символів. Майже кожен законний символ Unicode може з'являтися в документі XML. Процесор аналізує розмітку і передає структуровану інформацію додатку. Специфікація встановлює вимоги щодо того, що повинен робити і чого не повинен робити процесор XML, але програма не входить до сфери її застосування.

XML теги визначають структуру та значення даних користувача. Тег - це конструкція розмітки, яка починається з <і закінчується>. Теги бувають трьох смаків:

- стартовий тег, такий як <section>;
- кінцевий тег, наприклад </section>;
- тег порожнього елемента, наприклад <line-break />.

Використання XML в Android Studio значно полегшує роботу з додатком, а саме з його інтерфейсом, та допомагає запрограмувати всі елементи, які містяться в додатку.

Макет визначає структуру для користувальницького інтерфейсу у додатку. Усі елементи в макеті побудовані з використанням ієрархії об'єктів View та ViewGroup. View зазвичай малює те, що користувач може бачити та взаємодіяти. Тоді як ViewGroup - це невидимий контейнер, який визначає структуру макета для View та інших об’єктів ViewGroup, як показано на рисунку 2.5.

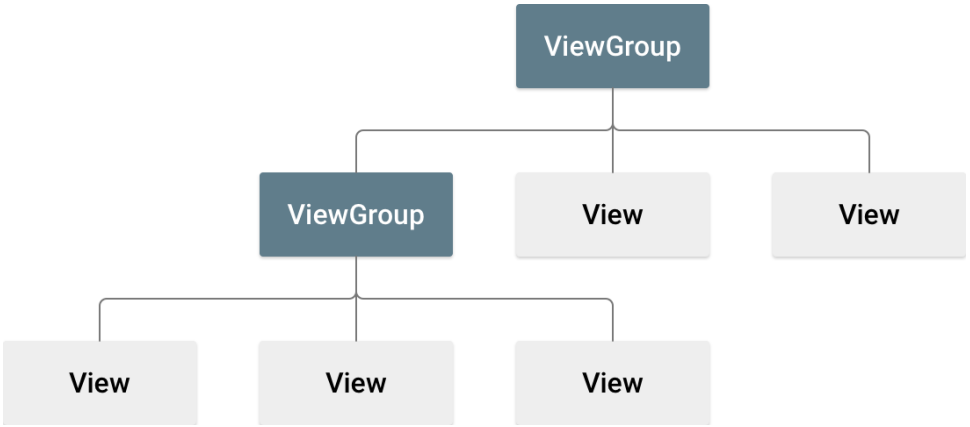


Рисунок 2.5 – Структура макету

Об'єкти View зазвичай називаються "віджетами" і можуть бути одним із багатьох підкласів, таких як Button або TextView. Об'єкти ViewGroup, як правило, називаються "макетами", можуть бути одним із багатьох типів, що забезпечують різну структуру макета, наприклад LinearLayout або ConstraintLayout.

Можливо оголосити макет двома способами:

- оголосити елементи інтерфейсу в XML;
- використати редактор макета Android Studio для створення XML-макета за допомогою інтерфейсу перетягування.

Екземпляри елементів макета під час виконання, додаток може програмно створювати об’єкти View та ViewGroup і керувати їх властивостями.

Оголошення інтерфейсу в XML дозволяє відокремити презентацію програми від коду, який контролює її поведінку. Використання XML-файлів також дозволяє легко надавати різні макети для різних розмірів екрана та орієнтації (обговорюється далі в розділі Підтримка різних розмірів екрана).

Фреймворк Android надає гнучкість використання одного або обох цих методів для створення інтерфейсу додатка. Наприклад, оголосити макети програми за замовчуванням у XML, а потім змінити макет під час виконання.

2.3.3 База даних SQLite

SQLite — це система управління реляційними базами даних (RDBMS), що міститься в бібліотеці C. На відміну від багатьох інших систем управління базами даних, SQLite не є механізмом баз даних клієнт-сервер. Швидше, це вбудовано в кінцеву програму.

SQLite - популярний вибір як програмне забезпечення для вбудованих баз даних для локального зберігання в прикладних програмах. Це найбільш широко розгортаний механізм баз даних, оскільки сьогодні він використовується серед багатьох широко розповсюджених браузерів, операційних систем та вбудованих систем (таких як мобільні телефони). SQLite має прив'язки до багатьох мов програмування, що дуже спрощує їхню роботу.

SQLite використовує незвичну систему типів для SQL-сумісної СУБД: замість присвоєння типу стовпцю, як у більшості систем баз даних SQL, типи присвоюються окремим значенням; в мовному відношенні він динамічно набирається.

Завдяки архітектурі двигуна (рисунок 2.6) можливо використання як на вбудованих системах, так і на виділених машинах з гігабайтними масивами даних.

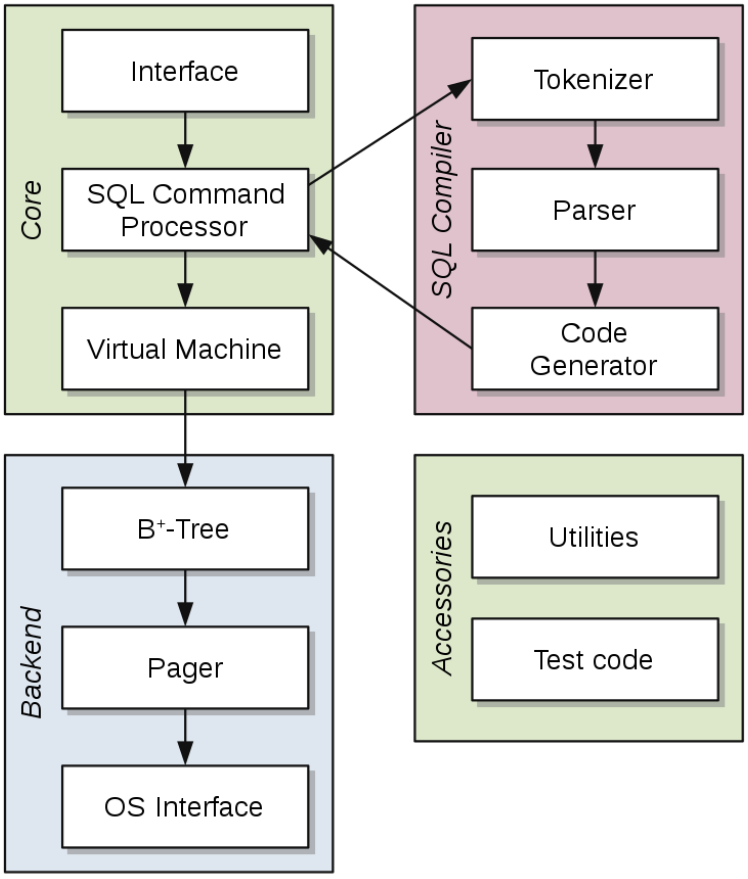


Рисунок 2.6 – Архітектура SQLite

Головні особливості бази даних:

- встановлення без конфігурації;
- база даних зберігається в одному крос-платформовому файлі на диску;
- швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій;
- простий, легкий у використанні API;
- автономність: немає зовнішніх залежностей;
- крос-платформовість, легко переноситься на інші системи.

Збереження даних у базі даних ідеально підходить для повторення або структурованих даних, таких як контактна інформація. API, які знадобляться для використання бази даних на Android, доступні в пакеті android.database.sqlite.

2.3.4 Вибір архітектури ПЗ для програми

Архітектура програмного забезпечення відноситься до фундаментальних структур програмної системи та дисципліни створення таких структур і систем. Кожна структура містить програмні елементи, відносини між ними та властивості як елементів, так і відносин. Функціонує як проект системи та проекту, що розробляється, викладаючи завдання, необхідні для виконання проектними групами.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) додаток розділяється на три основні логічні компоненти: модель, вигляд та контролер. Кожен із цих компонентів створений для обробки конкретних аспектів розробки програми. MVC - одна з найбільш часто використовуваних галузевих стандартів веб-розробки для створення масштабованих та розширюваних проектів.

Шаблон моделі (рисунок 2.7) поділяється на такі компоненти:

- компонент Model відповідає всій логіці, пов’язаній з даними, з якою працює користувач. Це може представляти або дані, які передаються між компонентами View та Controller, або будь-які інші дані, пов’язані з бізнес-логікою. Наприклад, об'єкт "Клієнт" буде отримувати інформацію про клієнта з бази даних, маніпулювати нею та оновлювати дані назад у базу даних або використовувати її для рендерингу даних;
- компонент View використовується для всієї логіки інтерфейсу програми. Наприклад, подання Клієнт включатиме всі компоненти інтерфейсу, такі як

текстові поля, випадаючі меню тощо, з якими взаємодіє кінцевий користувач;

- контролери виступають інтерфейсом між компонентами Model і View для обробки всієї бізнес-логіки та вхідних запитів, маніпулювання даними за допомогою компонента Model та взаємодії з поданнями для надання кінцевого результату. Наприклад, контролер замовника буде обробляти всі взаємодії та входи з перегляду замовника та оновлювати базу даних за допомогою моделі замовника. Той самий контролер буде використовуватися для перегляду даних Клієнта.

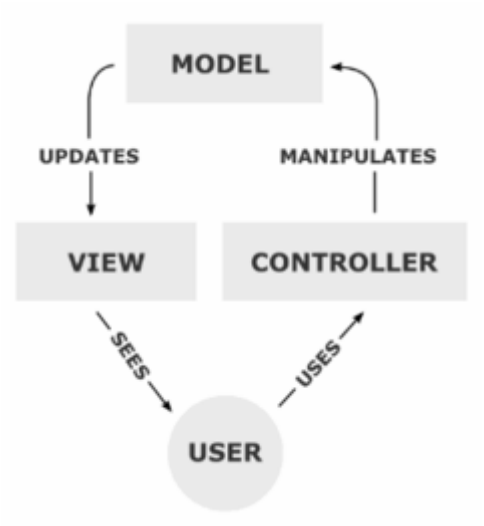


Рисунок 2.7 – Шаблон MVC

Модель–Представлення–Пред'явник (*MVP*) — шаблон проєктування, є похідним від архітектурного шаблону модель-вигляд-контролер (MVC) і використовується в основному для побудови інтерфейсів користувача.

Шаблон відділяє візуальне відображення та поведінку обробки подій у різні класи (рисунок 2.8), а саме: Представлення (*View*) та Пред'явник (*Presenter*).

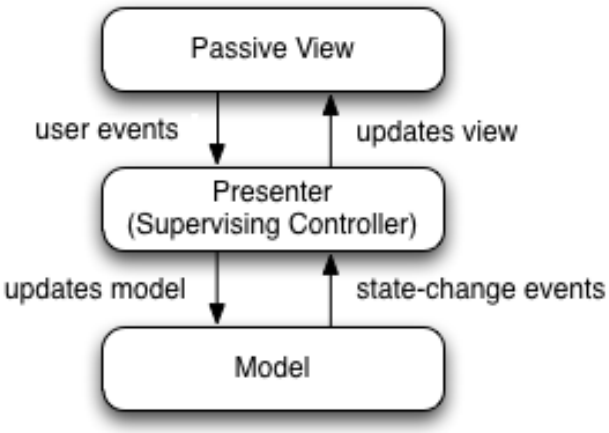


Рисунок 2.8 – Шаблон MVP

MVP - це архітектурний шаблон інтерфейсу користувача, розроблений для полегшення автоматизованого модульного тестування та покращення поділу проблем у логіці презентації.

Шаблон (рисунок 2.9) поділяється на такі частини:

- модель - інтерфейс, що визначає дані, які відображатимуться або користуватимуться іншим способом в інтерфейсі користувача;
- представлення - пасивний інтерфейс, який відображає дані (модель) і направляє команди користувача (події) до пред'явника, щоб діяти на основі цих даних;
- пред'явник - діє на модель та представлення. Він отримує дані зі сховищ (модель) і форматує їх для відображення у поданні.

Архітектура проєктування під назвою Model, View, ViewModel (MVVM) – програмний архітектурний шаблон, який полегшує відокремлення розвитку графічного інтерфейсу користувача, за допомогою мови розмітки або коду графічного інтерфейсу, від розвитку бізнес-логіки або зворотної кінцевої логіки (модель), щоб подання не залежало від якоїсь конкретної платформи моделі.

Модель подання MVVM є перетворювачем значень, що означає, що модель подання відповідає за перетворення об'єктів даних із моделі таким чином, що об'єкти легко управляти та представляти. У цьому відношенні View Model більше є моделлю, ніж подання, і обробляє більшість, якщо не всю логіку відображення подання. Модель подання може реалізовувати шаблон посередника, організовуючи доступ до внутрішньої логіки навколо набору випадків використання, підтримуваних видом. MVVM була створена з метою поділу праці дизайнера і програміста.

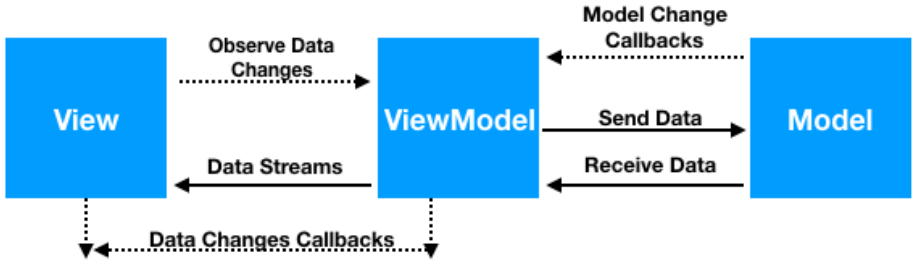


Рисунок 2.9 – Шаблон MVVM

MVVM зручно використовувати замість класичного MVC та йому подібних у тих випадках, коли на платформі, присутнє «зв'язування даних».

MVVM був розроблений для видалення практично всього графічного коду графічного інтерфейсу з рівня представлення даних, використовуючи функції прив'язки даних у WPF, щоб краще полегшити відділення розвитку рівня подання від решти шаблону. Шаблон MVVM намагається отримати обидві переваги розділення функціонального розвитку, що надаються MVC, одночасно використовуючи переваги прив'язки даних та фреймворку, прив'язуючи дані якомога ближче до чистої моделі програми.

Шаблон поділяється на 3 частини:

- модель – відноситься або до моделі домену, яка представляє вміст реального стану (об'єктно-орієнтований підхід), або до рівня доступу до даних, який представляє вміст (підхід, орієнтований на дані);
- подання – структура, макет та вигляд того, що користувач бачить на екрані;
- модель подання – абстракція виду, що відкриває загальнодоступні властивості та команди. Замість контролера шаблону MVC або презентатора шаблону MVP, MVVM має підшивку, яка автоматизує зв'язок між видом та його пов'язаними властивостями в моделі представлення.

Проаналізувавши всі наведені шаблони архітектури програмного забезпечення для створення додатку «Events» доцільніше використовувати модель MVVM.

3. РОЗРОБКА ANDROID-ДОДАТКУ ДЛЯ ПОШУКУ ТА ДОДАВАННЯ ЗАХОДІВ

3.1 Основні компоненти додатку

Зробивши детальний аналіз потрібно зауважити, що розроблюваний додаток Events написаний в середовищі розробки Android Studio використовуючи мову програмування Java, засоби мови XML, влаштовану базу даних SQLite і сервіси Google Maps та функції геолокації та геокодування, а також застосовує архітектуру проєктування MVVM.

В додатку Events доступні Activity, які знаходяться в каталозі app/java/com.example.events (рис. 3.1):

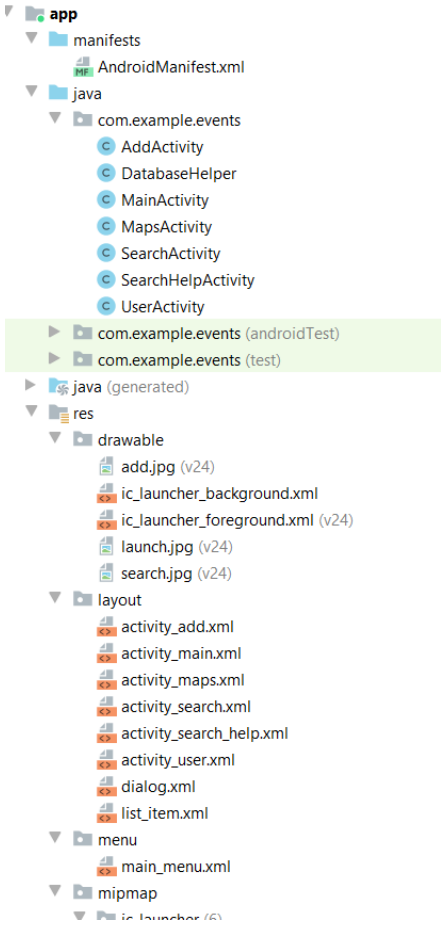


Рисунок 3.1 - Складові програми в середовищі розробки

MainActivity – головний екран програми, який містить головний інтерфейс та доступ через кнопки до інших Activity, а також доступ до меню, яке

викликається натиском справа вгорі на 3 крапки, що відкриває доступ користувачу до налаштувань та інформації про додаток, що визначено в цій діяльності так:

```
Dialog dialog = new Dialog(MainActivity.this);
dialog.setContentView(R.layout.dialog);
TextView text = (TextView) dialog.findViewById(R.id.dialogText);
switch(id){
  case R.id.settings :
    dialog.setTitle("Settings");
    text.setText("This section is in development");
    dialog.show();
    return true;
  case R.id.about:
    dialog.setTitle("About app");
    text.setText("This app author is Pankov Tymur\nEvents created for adding and searching events\nAnd also for bachelor`s degree");
    dialog.show();
    return true;
}
```

AddActivity – діяльність, в якій представлена можливість додання даних до існуючої бази даних нових заходів зі зручним інтерфейсом обрання та вводу даних. Для додавання застосовуються Content Values та функції put та insert, передбачені бібліотекою sqlite:

```
ContentValues cv = new ContentValues();
cv.put(DatabaseHelper.COLUMN_NAME, nameBox.getText().toString());
cv.put(DatabaseHelper.COLUMN_PHONE,
Integer.parseInt(phoneBox.getText().toString()));
cv.put(DatabaseHelper.COLUMN_PLACE, placeBox.getText().toString());
```

```
cv.put(DatabaseHelper.COLUMN_DATE, dateBox.getText().toString());
cv.put(DatabaseHelper.COLUMN_INFO, infoBox.getText().toString());
db.insert(DatabaseHelper.TABLE, null, cv);
goHome();
```

Для зручного обрання дати та часу застосовувались елементи TimePicker та DatePicker:

```
TimePickerDialog.OnTimeSetListener t=new
TimePickerDialog.OnTimeSetListener() {
  public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
    dateAndTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
    dateAndTime.set(Calendar.MINUTE, minute);
    setInitialDateTime();
  }};
DatePickerDialog.OnDateSetListener d=new
DatePickerDialog.OnDateSetListener() {
  public void onDateSet(DatePicker view, int year, int monthOfYear, int
dayOfMonth) {
    dateAndTime.set(Calendar.YEAR, year);
    dateAndTime.set(Calendar.MONTH, monthOfYear);
    dateAndTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);
    setInitialDateTime();
  }};
public void setDate(View v) {
  new DatePickerDialog(AddActivity.this, d,
      dateAndTime.get(Calendar.YEAR),
      dateAndTime.get(Calendar.MONTH),
      dateAndTime.get(Calendar.DAY_OF_MONTH))
      .show();
```

```
    }
    public void setTime(View v) {
      new TimePickerDialog(AddActivity.this, t,
          dateAndTime.get(Calendar.HOUR_OF_DAY),
          dateAndTime.get(Calendar.MINUTE), true)
          .show();
    }
    private void setInitialDateTime() {
      dateBox.setText(DateUtils.formatDateTime(this,
          dateAndTime.getTimeInMillis(),
          DateUtils.FORMAT_SHOW_DATE |
    DateUtils.FORMAT_SHOW_YEAR
              | DateUtils.FORMAT_SHOW_TIME));
    }
```

MapsActivity – діяльність, яка передбачає функції геолокації та геокодування, а також відображення місця на мапах Google та можливість побудови шляху до нього. Основні засоби, які використовуються для цього в програмі:

```
// Add a marker in place and move the camera
LatLng place = new LatLng(latitude,longitude);
mMap.addMarker(new MarkerOptions().position(place).title("It`s your place of event"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(place));
```

SearchActivity – діяльність, яка дозволяє переглянути наявні заходи в базі даних з функцією пошуку та відкриття детальної інформації по натиску на окремий захід. В програмі натиск та визначення заходу виконується за допомогою List та Filter:

```
userList = (ListView) findViewById(R.id.list);
userFilter = (EditText)findViewById(R.id.userFilter);
// при нажатии на list
userList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
  @Override
  public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
    Intent intent = new Intent(getApplicationContext(),
SearchHelpActivity.class);
    intent.putExtra("id", id);
    startActivity(intent);
  }
});
```

Підключення до бази даних та отримання даних виконується завдяки Cursor:

```
db = databaseHelper.getReadableDatabase();
userCursor = db.rawQuery("select * from " + DatabaseHelper.TABLE, null);
String[] headers = new String[]{DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_DATE};
userAdapter = new SimpleCursorAdapter(this,
android.R.layout.simple_list_item_2,
  userCursor, headers, new int[]{android.R.id.text1, android.R.id.text2}, 0);
```

SearchHelpActivity – діяльність, яка показує детальну інформацію, що наявна в базі даних про окремий захід та дозволяє перейти до MapsActivity. Для показу інформації відбувається безпосередня взаємодія з базою даних для отримання id елемента і відображення інформації по ньому:

```
Bundle extras = getIntent().getExtras();
userId = extras.getLong("id");
// получаем элемент по id из бд
```

```
userCursor = db.rawQuery("select * from " + DatabaseHelper.TABLE + "
where " +
    DatabaseHelper.COLUMN_ID + "=?", new
String[]{String.valueOf(userId)});
userCursor.moveToFirst();
nameBox.setText("Event`s name\n\t"+userCursor.getString(1));
phoneBox.setText("Supervisor`s phone
\n\t+380"+String.valueOf(userCursor.getInt(2)));
placeBox.setText("Place of event:\n\t"+userCursor.getString(3));
dateBox.setText("Date of the event\n\t"+userCursor.getString(4));
infoBox.setText("Information about event:\n\t"+userCursor.getString(5));
userCursor.close();
```

DatabaseHelper – діяльність, яка є допоміжною в роботі з базою даних SQLite (початкова база, ідентифікація елементів, пошук).

Програма має в наявності Intents (Наміри).

Наміри задіяні для переходу між екранами інтерфейсу та запусками сервісів додатку таким чином:

```
Intent intent = new Intent(this, AddActivity.class);
startActivity(intent);
```

Програма містить головний системний файл – AndroidManifest.xml, де оголошені всі служби та сервіси, зокрема для роботи з мапами Google в цей файл додавалися оголошення та ключі. Основні з них представлені таким чином у програмі:

```
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
```

```
android:supportsRtl="true"
android:theme="@style/Theme.Events">
                    <meta-data
   android:name="com.google.android.geo.API_KEY"
android:value="AIzaSyDN9RtK8JPQof5eZpgupAmOmAg8deErQPo"
/>
                    <activity
   android:name=".MapsActivity"
   android:label="@string/title_activity_maps"></activity>
                    <activity     android:name=".SearchHelpActivity"     />
<activity                  android:name=".SearchActivity"                 />
<activity                   android:name=".AddActivity"                   />
<activity                   android:name=".UserActivity"                  />
<activity android:name=".MainActivity">
                    <intent-filter>
      <action         android:name="android.intent.action.MAIN"         />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

Елементи, що відповідають за зображення, іконки додатку, кольори та подібні елементи інтерфейсу зберігаються в каталогах res/mipmap та res/values (рисунок 3.2):

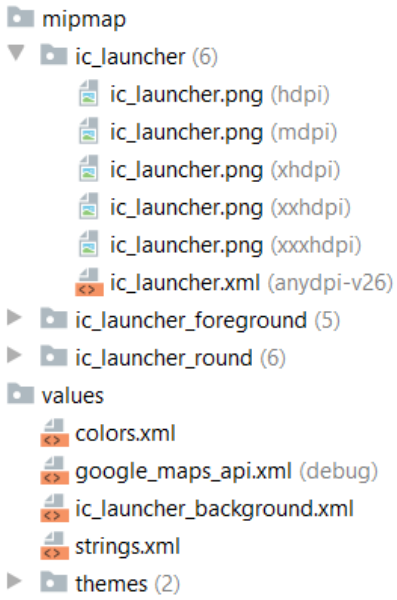


Рисунок 3.2 – Каталоги res/mipmap та res/values

Каталог «res.drawable» зберігає в собі файли, які використовуються в програмі, наприклад, зображення в середині додатку.

Каталог «res.layout» зберігає в собі файли xml, які потрібні для графічного відображення інтерфейсу кожної Activity.

Каталог «res.menu» зберігає в собі файли xml, які потрібні для графічного відображення меню (в додатку Events справа вгорі) в кожній Activity.

Каталог «res.mipmap.ic_launcher» зберігає в собі файли, які використовуються для коректного відображення значка додатку на пристроях.

Системні файли, такі як build.gradle, зберігаються в окремому каталозі Gradle Scripts (рисунок 3.3).

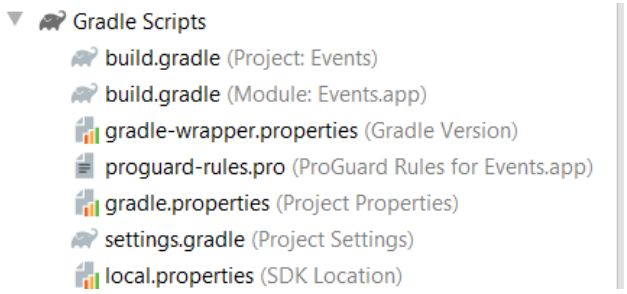


Рисунок 3.3 – Каталог Gradle Scripts

Системний файл build.gradle дозволяє вносити зміни для використання не стандартних сервісів, таких як Google Maps. Всі основні підключення

прописуються в dependencies, в програмі присутні такі:

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.android.gms:play-services-maps:17.0.1'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

В каталозі res/layout зберігаються файли типу xml, які визначають загальний вигляд Activity в додатку, а також два додаткових файли, які визначені для користування в середині цих діяльностей. Android Studio дозволяє змінювати інтерфейс користувача в декілька способів:

- прописати в коді xml;
- зробити інтерфейс мануально;
- змішаний спосіб;

Під час розробки програми був використаний змішаний спосіб, так як він здається найбільш зручним, адже деякі елементи зручніше запрограмувати, а деякі додати за допомогою миші. В коді елементів передбачається багато визначень, такі як id, висота, ширина, розташування відносно інших елементів (якщо присутні), текст та його розмір та інші. Наприклад, в додатку при додаванні заходу кнопка для обрання дати має такі параметри:

```
<Button
    android:id="@+id/dateButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
```

```
android:layout_marginTop="12dp"
android:onClick="setDate"
android:text="Change                                    date"
app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toBottomOf="@id/timeButton" />
```

3.2 База даних додатку

Додаток має власну влаштовану базу даних SQLite. Рушій мови не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. що дає змогу зручно користуватися нею та коригувати для своїх потреб у власній розробці. Саме це зумовило використання в даному проєкті.

Сама база даних має такі параметри:

- унікальний номер id (не видно користувачу);
- назва заходу;
- телефон організатора;
- місце проведення;
- дата та час;
- детальна інформація.

В програмі їх реалізовано в окремому файлі таким чином:

```
private static final String DATABASE_NAME = "eventsdb.db"; // назва бд
private static final int SCHEMA = 1; // версія базы данных
static final String TABLE = "events"; // назва таблиці в бд
```

```
// назва параметрів
public static final String COLUMN_ID = "_id";
public static final String COLUMN_NAME = "name";
public static final String COLUMN_PHONE = "phone";
public static final String COLUMN_PLACE = "place";
public static final String COLUMN_DATE = "date";
public static final String COLUMN_INFO = "info";
```

Для його визначення та додавання в методі onCreate було прописано:

```
db.execSQL("CREATE TABLE users (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + COLUMN_NAME + " TEXT, " + COLUMN_PHONE + " INTEGER, " + COLUMN_PLACE + " TEXT," + COLUMN_DATE + " TEXT," + COLUMN_INFO + " TEXT);");
// додавання початкових даних
db.execSQL("INSERT INTO "+ TABLE +" (" + COLUMN_NAME + ", " + COLUMN_PHONE + ", " + COLUMN_PLACE + ", " + COLUMN_DATE + ", " + COLUMN_INFO + ") " +
"VALUES ('Diploma', 0990197820, 'Sidney', 'June 8, 2021, 8:00 AM', 'Defence diploma" + "Pankov Tymur" + "KV 72');");
db.execSQL("INSERT INTO "+ TABLE +" (" + COLUMN_NAME + ", " + COLUMN_PHONE + ", " + COLUMN_PLACE + ", " + COLUMN_DATE + ", " + COLUMN_INFO + ") " +
"VALUES ('Home party', 0502563785, 'Obolonsky, 27', 'May 30, 2022, 8:00 PM', 'Party for my friends about my graduate');");
```

3.3 Інтерфейс користувача

Інтерфейс користувача для додатка Android будується за допомогою ієрархії макетів (об'єктів ViewGroup) та віджетів (об'єктів View). Макети - це невидимі контейнери, які контролюють, як розташовані на екрані створені вигляди.

Віджети - це такі компоненти інтерфейсу користувача, як кнопки та текстові поля.

Початковий екран представлений на рисунку 3.4. Передбачається обрання бажаної функції – додавання власного заходу за допомогою кнопки «ADD EVENT» або пошук існуючого заходу за допомогою кнопки «SEARCH EVENT».

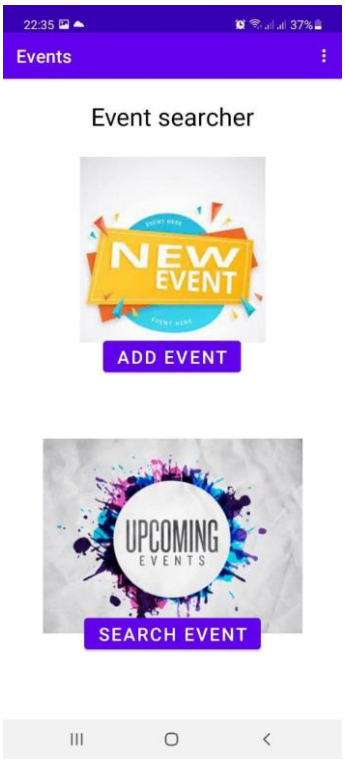


Рисунок 3.4 - Початковий екран

По натиску на 3 крапки вгорі справа відкривається меню з додатковим функціоналом (рисунок 3.5).

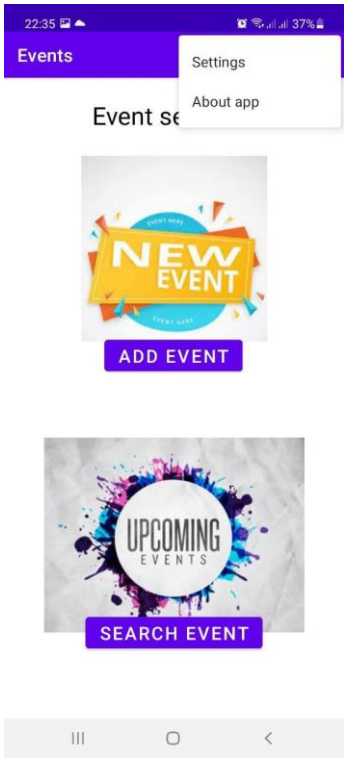


Рисунок 3.5 – Меню з додатковим функціоналом

При натисканні на кнопку «About app» з’являється діалогове вікно з додатковою інформацією (рисунок 3.6).

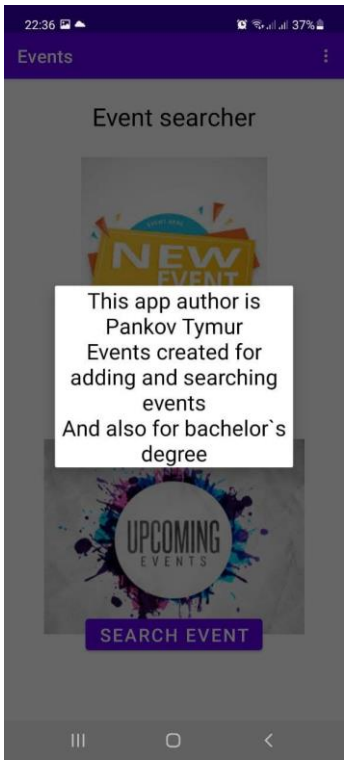


Рисунок 3.6 – Додаткова інформація про додаток

Після переходу по кнопці «ADD EVENT» відображається нова Activity (рисунок 3.7), яка дозволяє додати до бази даних додатку новий захід з інформацією про нього. Для відміни додавання заходу потрібно просто натиснути кнопку «назад».

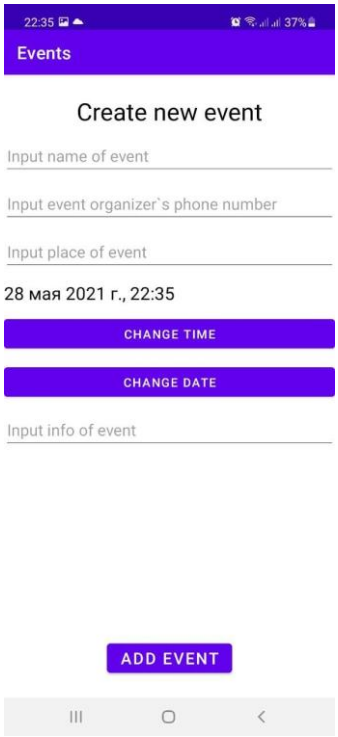


Рисунок 3.7 - Додавання нового заходу

При натисканні на кнопку «CHANGE TIME» відкривається Time Picker для обрання часу початку заходу (рисунок 3.8). Також присутня функція вводу часу вручну (значок клавіатури).

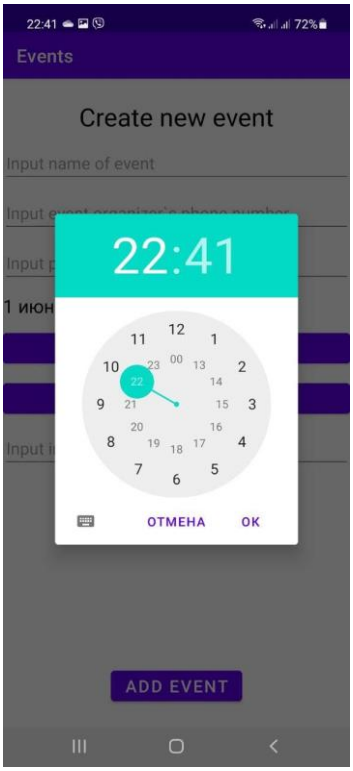


Рисунок 3.8 – Time Picker

При натисканні на кнопку «CHANGE DATE» відкривається Date Picker для обрання дати заходу (рисунок 3.9).

Рисунок 3.9 – Date Picker

По натиску кнопки «SEARCH EVENT» відкривається Activity, що дозволяє переглянути існуючі заходи (рисунок 3.10). Також представлена можливість пошуку по базі даних.

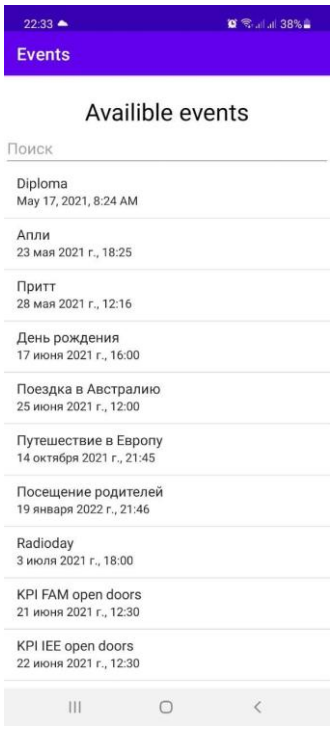


Рисунок 3.10 - Доступні заходи в базі даних

При натиску на один з елементів списку в Activity з пошуку заходів здійснюється перехід до наступної діяльності, яка передбачає показ детальної інформації про обраний (рисунок 3.11).

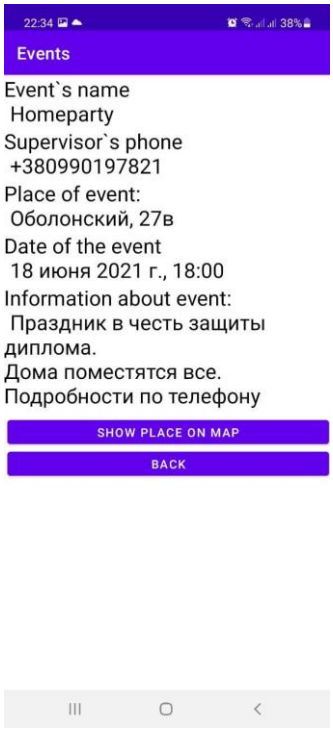


Рис. 3.11 - Детальна інформація про захід

Якщо натиснути на кнопку «SHOW PLACE ON MAP» відкривається Activity взаємодії з мапами та показує місце розташування даної події (рисунок 3.12).

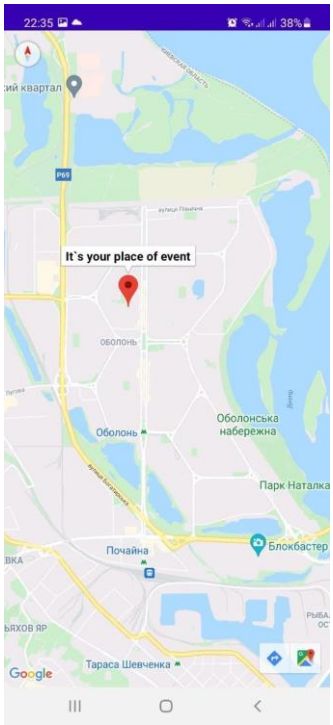


Рисунок 3.12 – Розташування обраного заходу на мапі

ВИСНОВКИ

Мета даного дипломного проєкту – розробка Android-додатку Events мовою програмування Java.

В ході роботи був проведений детальний аналіз існуючих альтернативних рішень проблеми пошуку заходів та інструментів розробки програм і визначено, що доцільно використовувати існуюче середовище розробки Android Studio від JetBrains, створене спеціально для розробки додатків під операційну систему Android, додаток має мати зручний та зрозумілий інтерфейс користувача, який можливо зробити за допомогою мови XML, представленої в Android Studio, програма має мати влаштовану базу даних SQLite через зручність влаштування та оперування всередині програми, а також для можливості коректної роботи програми в режимі офлайн, продукт повинен мати вбудовані мапи Google та використовувати функції геолокації та геокодування для зручності роботи з пошуком шляху до обраного заходу користувачем, в додатку має бути представлена функція пошуку по базі даних легкості вибору потрібного заходу.

Розроблений додаток надає користувачам максимально зручне рішення питання проєкту для користування у власних цілях. Для розробників проєкт може бути цікавим в плані рішень використання влаштованої бази даних SQLite та оперування даними всередині програми, а також використання сервісу Google Maps та функції геокодування.

В проєкт додатку закладено можливість подальшої модифікації для покращення функціоналу та розширення для покриття більшої території та розповсюдження.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Програмування під Android на Java: URL: https://metanit.com/java/android/ (дата звернення: січень-лютий 2021).
2. Maps SDK for Android: URL: https://developers.google.com/maps/documentation/android-sdk/map-with-marker?hl=ru (дата звернення: березень-квітень 2021).
3. Google Maps API в android: URL: https://habr.com/ru/post/341548/. (дата звернення: березень 2021).
4. Робота з базою даних SQLite: URL: http://developer.alexanderklimov.ru/android/sqlite/android-sqlite.php (дата звернення: лютий-березень 2021).
5. Теоретичні відомості по мові програмування Java: URL: https://uk.wikipedia.org/wiki/Java (дата звернення: квітень 2021).
6. Реалізація паттерна MVVM: URL: https://habr.com/ru/post/176867/ - (дата звернення: квітень 2021).
7. Реалізація паттерна MVP: URL: https://startandroid.ru/ru/blog/493-mvp-dlja-nachinajuschih-bez-bibliotek-i-interfejsov.html (дата звернення: квітень 2021).
8. Реалізація паттерна MVC: URL: https://javarush.ru/quests/lectures/questcollections.level06.lecture01 (дата звернення: квітень 2021).
9. Вбудована база даних SQLite: URL: https://habr.com/ru/post/149356/ (дата звернення: квітень 2021).
10. Завантаження середовища розробки Android Studio: URL: https://developer.android.com/studio (дата звернення: вересень 2020).
11. Додавання та налаштування сервісу Google Maps в Android Studio: URL: https://developers.google.com/maps/documentation/android-sdk/map-with-marker?hl=ru (дата звернення: березень 2021).

12. Робота з інтерфейсом в Google Maps SDK для Android: URL: https://habr.com/ru/post/484100/ (дата звернення: березень 2021).
13. Шаблони Google Maps Activity: URL: http://developer.alexanderklimov.ru/android/google_maps.php (дата звернення: березень 2021).
14. Додаток AroundMe: URL: https://play.google.com/store/apps/details?id=com.tweakersoft.aroundme&hl=ru&gl=US (дата звернення: січень 2021.
15. Додаток Eventbrite: URL: https://play.google.com/store/apps/details?id=com.eventbrite.attendee&hl=ru&gl=US (дата звернення: січень 2021).
16. Java Activity: URL: https://developer.android.com/reference/android/app/Activity (дата звернення: жовтень 2020).

ДОДАТКИ